(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 041 500 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
04.10.2000 Bulletin 2000/40

(51) Int. Cl.[7]: G06F 17/60

(21) Application number: 00301916.3

(22) Date of filing: 08.03.2000

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: 31.03.1999 GB 9907477

(71) Applicant:
Nortel Networks Limited
Montreal, Quebec H2Y 3Y4 (CA)

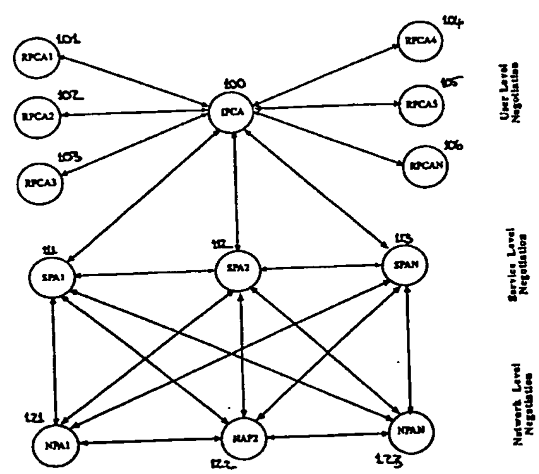
(72) Inventors:
- Buckle, Philip
  Ware, Hertfordshire SG12 0QB (GB)
- Hadingham, Robert
  Harlow, Essex CM19 5RP (GB)
- Jennings, Nicholas Robert
  Southampton SO32 2LS (GB)
- Faratin, Peyman
  London SW10 9DR (GB)

(74) Representative:
Ryan, John Peter William
Nortel Networks
Intellectual Property Law Group
London Road
Harlow, Essex CM17 9NA (GB)

(54) Flexible agent-based negociators

(57) A method for agent negotiation over a set of issues, between automated entities comprising the step of at least one of the entities making a counter-offer wherein its score for at least one of said issues is lowered and its score for at least one of said issues is raised. The method can be further used for negotiation over a set of issues comprising at lest one qualitative issue the values of the qualitative issues are mapped to qualitative values which are then used in the negotiation.

Fig. 1



A3

EP 1 041 500 A2

Printed by Xerox (UK) Business Services
2.16.7 (HRS)/3.6

Description

FIELD OF THE INVENTION

[0001] The present invention relates to a method and apparatus for a performing automated agent-based negotiation and a system incorporating the same.

BACKGROUND TO THE INVENTION

[0002] Automated agents are autonomous entities which decide for themselves what, when, and under which conditions their actions should be performed. Since the agents have no direct control over others, they must negotiate with others in order to cause them to act in a particular manner. Negotiations are processed by which a joint decision is made by two or more parties. The parties first verbalise potentially contradictory demands and then move towards agreement. Negotiating agents may populate different types of environments which may require either a very simple and responsive decision to be made (for example buying and selling of goods), or a complex and deliberative problem solving activities (e.g. planning), or a combination of both. Negotiation decisions may therefore be viewed as composed of responsive and/or deliberative components. The outcome of these decisions can result in either concession or search for new alternatives.

[0003] In the past formal models of choice have achieved co-ordination through specification of the negotiation space comprising the issues which agents negotiate over, along with their possible values, which determine the set of alternative solutions. Negotiation is then considered as an optimisation problem where, given the utility function of the agents, the best solution is sought. Such formal models often ignore interactions, and involve unrealistic assumptions. These assumptions may include an assumption that the agents share common knowledge and each have an unlimited computational capability. Interactions are viewed in these models as unnecessary since rational and super-logical agents can reach agreements instantly given the common knowledge and unlimited computational power assumptions.

[0004] An alternative approach to co-ordination is to specify the rules (or order) of interaction, detailing which agents can say what and when. Absence of any normative rule of behaviour may lead to unmanageable interactions. An example of such normative rules is to be found in the Contract Net Protocol of Davies & Smith 1998.

[0005] In addition to being provided with an interaction protocol, agents must also be provided with the capability to represent and reason about, within their information and resource bounds, both their internal and their external world view and with the capacity to interact according to the protocol.

[0006] "Faratin, Sierra, & Jennings 1998" gives an in-draft explanation of evaluatory and responsive mechanisms.

OBJECT TO THE INVENTION

[0007] The invention seeks to provide an improved method and apparatus for performing automated negotiation.

SUMMARY OF THE INVENTION

[0008] According to a first aspect of the present invention there is provided a method of negotiation, over a set of issues, between automated entities comprising the step of:

modifying the set of issues during the negotiation.

[0009] According to a second aspect of the present invention there is provided a method of negotiation over a set of issues, between automated entities comprising the step of:

at least one of the entities making a counter-offer wherein its score for at least one of said issues is lowered and its score for at least one of said issues is raised.

[0010] According to a third aspect of the present invention there is provided a method of negotiation over a set of issues comprising at lest one qualitative issue, and comprising the steps of:

mapping the values of said qualitative issues to qualitative values;
negotiating using said qualitative values.

[0011] The invention also provides for a system for the purposes of digital signal processing which comprises one or more instances of apparatus embodying the present invention, together with other additional apparatus.

[0012] The invention also provides software on a machine-readable medium embodying the methods of the present invention.

[0013] The preferred features may be combined as appropriate, as would be apparent to a skilled person, and may be combined with any of the aspects of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] In order to show how the invention may be carried into effect, embodiments of the invention are now described below by way of example only and with reference to the accompanying figures in which:

Figure 1 shows an example of negotiation structure in accordance with the present invention;

Figure 2 shows an example of a negotiation protocol in accordance with the present invention;

Figure 3 shows a second view of a negotiation protocol in accordance with the present invention;

Figure 4 shows an example of mapping qualitative issues to a value system in accordance with the present invention.

DETAILED DESCRIPTION OF INVENTION

[0015] A first example embodiment is based on the use of negotiations to co-ordinate the dynamic provisioning of resources for a Virtual Private Network (VPN) for end users. This service is provided to the users by service and network providers. The arrangement is made up of a number of agents which represent the users, service and network providers.

[0016] Referring now to figure 1, users are represented by user agents, which are collectively referred to, as Personal Communication Agents, or PCAs. The PCA are either Initiating Personal Communication Agents, (IPCAs) 100 representing the user who has the desire to initiate the meeting, or Receiving Personal Communication Agents (RPCAs) 101-106, representing the party/parties who are required to attend the meeting, respectively. The interactions between these PCAs can be multilateral (involving one IPCA and multiple RPCAs) and are centred around negotiation over meeting scheduling, where each agent negotiates on behalf of its user and where the goal is to establish the most appropriate time and security level for the service requested by the IPCA. The set of issues over which PCA agents negotiate consists of Service-Type, Security, Price, Start-Time, and Duration, where Service-Type denotes the choice of the service (eg. Video, audio or mixture thereof), Price is the share of the price the agents should pay for the service, Start-Time is the time the service will commence and Duration is the length of the service respectively. The Security issue encodes the privacy of the meeting and is represented by both the method of security (e.g. in the order of value to PCAs, Entrust, Verisign or Microsoft) and the level of security method (again in the order of value, confidentiality, integrity and confidentiality).

[0017] IPCA and RPCA requirements are constrained by what resources are available at the network domain level. For example, the network may be heavily loaded at the time the service is required by the PCAs. Since the network is only visible to the IPCA through the Service Provider Agents (SPAs) 111-113, the thread of IPCA and RPCAs negotiation is executed in parallel with negotiation between IPCA and SPAs. The interactions between IPCA and SPA directly influence the meeting scheduling negotiations between IPCA and RPCAs. In this example service level negotiation between IPCA and SPAs is assumed to be bilateral. However, each SPA agent can make agreements with IPCA for services and outsource these commitments by initiating negotiation with other SPAs for services. The set of issues in the negotiation between IPCA and SPAs is the same as the meeting scheduling negotiation thread between IPCA and RPCAs with an additional element Participants which is the list of users (represented by RPCAs) specified to be included in the requested service.

[0018] Either concurrently or after the service is provisioned between IPCA and SPA and the Network Provider Agents (NPAs) 121-123 which manage the infrastructure and low level aspects of the IP network. This thread of interaction is multilateral since each NPA manages only a subset of the IP network. Therefore the SPA must negotiate with a number of NPAs in order to secure resources for services it provides to IPCA. The set of issues in the thread of negotiation between SPA and NPAs is made up of the following elements: Quality-of-Service, Security, Participants, Price, Start-Time and Duration. Quality-of-Service (QoS) represents the "goodness" of the service from an agent's perspective. QoS may be composed of a number of sub issues such as the Bandwidth (the capacity of the link), the latency (the delay imposed by the network on packets), the jitter (the maximum time deviation acceptable during transmission), the availability (percentage of the time over which the service is required) and packet loss (percentage of the total packets lost during the lifetime of the provisioned service).

[0019] Negotiation, in the scenario described above exhibits the following characteristics.

[0020] Agents negotiate for services. Services have a number of features/issues associated to them (e.g. their Price, Duration etc.), some of which can be dynamically introduced and retracted (eg. Qos), and successful negotiation involves resolution of these issues to the satisfaction of all parties involved.

[0021] Since agents are autonomous, the factors which influence their negotiation stance and behaviour are private and not available to other parties. Thus agents do not know what utilities their opponents place on various outcomes; they do not know what reasoning models they employ; they do not know their opponent's constraints; and they do not know whether an agreement is even possible at the outset (i.e. the participants may have non-intersecting ranges of acceptable outcomes).

[0022] Since plans and execution of services/activities are real time and dependent on one another, the provisioning process should respect the time and resource levels of the agents - negotiation should be responsive to the time and resource levels of the agent. For example, if the operating environment can afford it (in terms of time, resources, etc.) then an agent may decide to engage in complex deliberation procedures involving a more refined search of the space of possible outcomes. For instance, SPA and NPA agents can engage in costly computation and selection procedures for contracts that manipulate or trade off the set of issues involved in negotiation. Alternatively, as the environment changes (e.g. deadline to reach an agreement is approaching fast, resource usage for negotiation has reached some critical level, or the other agent is exhibiting a reluctance to change its offer, etc.) then one or both of the agents may begin to adopt a more responsive attitude towards their environment by conceding. Thus responsive behaviours are similar to reactive behaviours which consider environmental conditions and are simple and uncostly responses to the environment.

[0023] Co-ordinated behaviour during negotiation is enforced through the normative rules of the negotiation protocol. This example is restricted to bilateral negotiations but multilateral negotiations have been shown to be equivalent to a series of bi-lateral negotiations.

[0024] The protocol diagram of Figure 2 starts with a dialogue 201 to establish the conditions for negotiation (deadline, initial issues, etc.) Then, one of the agents makes an offer (transition from state 1 to state 2 or state 3) for contract ø. After that, the other agent can make a counter-offer as described below or a trade off (moving to state 2 or 3 depending on who started), and the agent that started the negotiation can in turn make a new counter-offer or a new trade off (going back to state 2 or 3). Since information models used by the agents are not publicly known (that is, agents do not know the reservation values of the other party over the negotiation issues), offers maybe outside the mutual zone of agreements. Therefore, agents may iterate between states 2 and 3 taking turns to offer new contracts. In either of these two states, one of the agents may accept the last offer made by the opponent thereby moving to state 4 or any of the agents may withdraw from the negotiation (moving to state 5). Agents withdraw from the negotiation process when the deadline of negotiation has been reached without reaching an agreement.

[0025] While at state 2 or 3 agents are permitted to start an elucidatory dialogue to establish a new set of issues to negotiate over. This protocol is a natural extension of the contract net protocol permitting iterated offer and counter-offer generation and permitting the modification of the set of issues under negotiation. Although neither termination nor convergence can be assured in the general case, in practice the existence of time deadlines ensures that the protocol will terminate.

[0026] Rational behaviour is assumed to consist of maximisation of a given value function (Raiffa 1982). Given this rationality stance, the decisions faced by agents in negotiation are often a combination of offer generation decisions (which initial offer should be generated, which counter offer should be given in situations where the opponent's offer is unacceptable), and evaluatory decisions (when negotiation should be abandoned, and when an agreement is reached). The solution to these decision problems is captured in the agent architecture. The components are mechanisms of the agent architecture which are responsible for generation of offers and counter offers are based on a distinction between mechanisms which are computationally uncostly and are responsive to the environment, and mechanisms which are relatively more costly because they engage in a more sophisticated search of the solution space.

[0027] The mechanisms which assist an agent with evaluation of offers is described next, followed by the generation mechanisms.

[0028] Evaluation of a contract consists of computing the value or score of the contract. When an agent, a, receives an offer x from agent b at time t, denoted by $x^t_{b \to a}$, over a set of issues J,

$$(x = (x[j_1], \ldots, x[j_n])$$

where $j_i \in J$), it rates the overall contract value, V, using the following weighted linear additive scoring function:

$$V^a(x) = \sum_{1 \le i \le n} w^a_j V^a_{ji}(x[ji])$$

[0029] Where w^a_{ji} is the importance (or weight) of issue ji such that

$$\sum_{1 \le i \le n} W^a{}_{ji} = 1$$

[0030] Given that the changing of the set of issues during negotiation is permitted, agents must be able to dynamically change the values of the weights. The score of value x[j] for agent a, given the domain of acceptable values Dj, is modelled as a scoring function $V^a_j : D_j \rightarrow [0,1]$. For convenience, scores are bounded to the interval [0,1], and the scoring functions are monotonic for quantitative issues. Given the score of the offered contract, the contract evaluation function determines whether to accept or reject the contract or generate a new contract to propose back to the other agent. Mechanisms which generate new contracts are presented below.

[0031] Known responsive mechanisms model reactive behaviours relative to a number of environmental factors. The underlying rationale and motivation of the design of these mechanisms is the need to model responsive behaviours responsive to growing environmental needs. For example, if an IPCA has committed lots of resources to its negotiation with SPAs and the time of a video service required by other RPCAs is approaching, then simple and less costly decision mechanisms which can result in concession may be preferred by IPCA.

[0032] Responsive mechanisms generate offers by linearly combining simple decay functions, called tactics. Tactics generate values for issues using only single environmental criteria. For example:

- **Time-dependent tactics** model increasing levels of concession as the deadline for the negotiation approaches.
- **Resource-dependent tactics** model increasing levels of concession with diminishing levels of resources, such as time.
- **Behaviour-dependent tactics** in which concession is based on the concessions of the other negotiating party.

[0033] However, to determine the best course of action agents may need to consider and assess more than just one environmental condition. Since each tactic generates a value for an issue using only a single criterion, the concept of strategy is introduced to model the modification, over time, of tactic weights as the criteria change their relative importance in response to environmental changes.

[0034] In addition to being responsive, Agents must also be deliberative. Two deliberative mechanisms are trade offs and issue set manipulations.

[0035] A trade off is a mechanism in which one party lowers its score on some issues and simultaneously demands more on other issues. For example, for the IPCA, offering a lower Price for a later Start-Time of a service may be equivalent in value (depending on the weights of the two issues) to offering a higher Price for an earlier Start-Time of a service. Thus, a trade off is a search for a contract that is equally valuable to the previous offered contract, but which may benefit the other party.

[0036] This decision mechanism is more costly than the responsive mechanisms because it involves searching all or a subset of possible contracts with the same score as the previously offered contract (hence there is no loss in contract utility) and selection of the contract which is the closest to the opponent's last contract offer.

[0037] Search is initiated by first generating new contracts which lie on what is called the iso-value (or indifference) curves (Raiffa 1982). Because all newly generated contracts lie on the same iso-value curve then agents are indifferent between any two given contracts on this curve.

[0038] Given a scoring value θ, the iso-curve set at degree θ for agent a is defined as:

$$iso_a(\theta) = \{x | V^a(x) = \theta\}$$

[0039] The selection of which contract to offer is then modelled as a "closeness function". Theory of fuzzy similarity is used in order to model "closeness". The best trade off then would be the most similar contract an the iso-curve.

[0040] Given an offer x, from agent a to agent b, and a subsequent offer y, from agent b to agent a, with $\theta = V^a(x)$, trade off for agent a with respect to y is defined as:

$$\text{tradeoff}_a(x,y) = \arg_x \max_{x \in \text{isoa}(\theta)} \{Sim(x,y)\}$$

[0041] Similarity between two contracts is defined as weighted combination of the similarity of the issues. Specifically the similarity between two contracts x and y over a set of issues J is defined as:

$$Sim(x,y) = \sum_{j \in J} w^a{}_j Sim_j(x[j],y[j])$$

where

$$\sum_{j \in J} w^a{}_j = 1$$

and where Sim_j is the similarity function for issue j.

[0042] Following the results from Valverde 1985, a similarity function (a function which satisfies the axioms of reflexivity, symmetry, and t-norm transitivity) can always be defined as conjunction (modelled as the infimum) of appropriate fuzzy equivalence relations induced by a set of criteria functions h_i. A criteria function is a function which maps from a given domain into values in [0,1]. For example, a function that models the criteria of whether a price is low, low price: Price → [0,1], could be defined as:

$$\text{Low price}(x) = \begin{cases} 1 & \text{if } x < £10 \\ \dfrac{£20-x}{£10} & \text{if } £10 < x < £20 \\ 0 & \text{if } x \geq £20 \end{cases}$$

[0043] Given a domain of values D_j, the similarity between two values $x,y \in D_j$, is defined as:

$$Sim_j(x,y) = \bigwedge_{1 \leq i \leq m} (h_i(x) \leftrightarrow h_i(y))$$

where $\{h_1, ..., h_m\}$ is a set of comparison criteria with $h_i : D_j \to [0,1]$, and $\leftrightarrow$ is an equivalence operator.

[0044] Simple examples of the equivalence operator $\leftrightarrow$, are

$$h(x) \leftrightarrow h(y) = 1 - |h(x) - h(y)|$$

or

$$h(x) \leftrightarrow h(y) = \min(h(y)/h(x), h(x)/h(y)).$$

[0045] Another deliberation mechanism is issue set manipulation. Negotiation processes are directed and centred around the resolution of conflicts over a set J of issues. This set may consist of just one or more issues (distributed and integrative bargaining respectively). For simplification the ontology of the set of possible negotiation issues J, is assumed to be shared knowledge amongst all the agents. It is further assumed that agents begin negotiation with a prespecified set of core issues, $J^{core} \subseteq J$, and possibly other mutually agreed non-core set members, $J^{\neg core} \subseteq J$. Alterations to J^{core} are not permitted since some features, such as the price of services may be mandatory. However, elements of $J^{\neg core}$ negotiation set may be altered dynamically. Agents can add issues to or remove issues from $J^{\neg core}$ as they search for new, previously unconsidered solutions.

[0046] In the scenario above agents negotiate over core packages. The negotiation between SPA and NPA agents

however consist of packages which comprise not only core issues but also include issues which can be added or removed throughout the process. For example, a SPA agent may begin QoS negotiation with a NPA agent, specifying only Bandwidth. However, NPA may subsequently decide to include into QoS negotiation a high packet loss issue if SPA has demanded a high capacity Bandwidth. Alternatively, SPA may remove the Bandwidth issue from QoS negotiation with NPA if IPCA has changed its demand from a high quality video service to a standard audio service.

[0047] If J^t is the set of issues being used at time t (where $J^t = \{j_1,...j_n\}$), and $J—J^t$ is the set of issues not being used at time t, and if $x^t = (x[j_1],..., x[j_n])$ is a's current offer to b at time t, then issue set manipulation may be defined through two operators, add and remove, which agents can apply to the set J^t. The add operator assists the agent in selecting an issue j' from $J—J^t$, and an associated value $x[j']$, which gives the highest score from the selecting agents perspective.

[0048] The best issue to add to the set J^t may be defined as:

$$Add(J^t) = \arg_{j max j \in J—Jt} \{\max_{x[j] \in Dj} V^a(x^t . x[j])\}$$

where . denotes concatenation.

[0049] An issue's score evaluation is also used to define the remove operator in a similar fashion to the add operator. This operator assists the agent in selecting the best issue to remove from the current negotiation set, J^t, with the highest score.

[0050] The best issue to remove from the set J^t (from a's perspective), may be defined as:

$$Remove(J^t) = \arg_{ji} \max_{ji \in Jt—Jcore} \{ V^a(x') \}$$

where

$$x' = (x^t[j_1],..., x^t[j_{i-1}], x^t[j_{i+1}], x^t[j_n])$$

[0051] The remove operator can also be defined in terms of the similarity function defined above. It selects from two given offers x (from agent a to b) and y (from agent b to a) which issue to remove from y so as to maximise the similarity with respect to y. We define this similarity based remove operator as:

[0052] The best issue to remove from a's perspective from the set J^t is defined as:

$$Remove(J^t) = \arg_{ji} \max_{ji \in J^t - J^{core}} \{ sim \; ((x[j_1],...,x[j_{i-1}], x[j_{i+1}],...,x[j_n]), (y[j_1],...,y[j_{i-1}], y[j_{i+1}],...,y[j_n])) \}$$

[0053] It is not possible to define a similarity based add operator since the introduction of an issue does not permit an agent to make comparisons with the opponent's last offer, because there is no value offered over that issue. Agents deliberate over how to combine these add and remove operators in a manner which maximises some measure —such as the contract score. However, a search of the tree of possible operators so as to find the optimum set of issues may be computationally expensive and require approximate and anytime algorithms. Another computational requirement of these mechanisms is the need for an agent to dynamically recompute the issue weights.

[0054] The protocol for establishing a new set of negotiating issues is isomorphic to the negotiation protocol described in Figure 2. The pre-negotiation phase is omitted (since the current set of issues have already been agreed). Ø is replaced by a new set of issues S, and primitives "propose" and "trade off" are replaced by primitive "new set" — a request for a new set of issues to be included into the negotiation. Each negotiating agent can start a dialogue over a new set of issues S (state 1 to state 2 or 3). Each agent can then either propose a new set (transition from state 2 to 3, depending on who started the dialogue), accept the other's proposed (state 4) set or withdraw (state 5).

[0055] The state transition arcs of figure 3 represent the participants utterances: → ⟨primitive⟩ are those of the servers and ⟨primitive⟩ → are those of the clients.

[0056] Negotiation is initiated when a client utters Call for Proposals cfp (state 31 to state 32). The server can then either indicate that it is capable (state 32 to 35) or that it is not (state 32 to failure). If the server has acknowledged its

capability or if the client knows it is capable because of information contained in its acquaintance model, the client may send out a proposal (state 33 to 34). The server can then either reject the proposal (state 34 to failure), accept the proposal (state 34 to 35) or counter-propose (state 34 to 36). If the server accepts, the client may either deny the contract to the server (state 35 to failure) or else confirm the contract (state 35 to success). Otherwise, if the server has counter-proposed (state 34 to 36) then the client may either accept the new contract (state 36 to 37), reject it (state 36 to failure) or else counter-propose a new contract (state 36 to 34). There may be several transitions between states 34 and 36. If it is the client who eventually accepts the contract (state 36 to 37), then the server may decide to either award the contract to the client (state 37 to success) or else deny it to that client (state 37 to failure).

[0057] The support of negotiation also requires a sound semantic specification of the communicative acts. The negotiation primitives described here consist of an initiator, propose, a reactor, counter-propose, two completers, accept and reject. In addition to these, two messages cfp, acknowledge and cannot are provided for the agents to set up a negotiation link.

cfp: The cfp interface enables agents to ask other agents whether they are able to provide a specified service, and whether they are prepared to negotiate for the provision of that service. Informally, the semantics of this message type is the question can you do this service?

- **(cfp** ⟨agent_id⟩ ⟨service_type⟩ ⟨conversion_id⟩ ⟨message_id⟩)

propose: The propose interface enables an agent to initiate negotiation by sending a proposal for the provision of a service, or receive a proposal from some other agent. The proposal consists of a set of attributes that specify the contents of an SLA. Although the type of a proposal is a list of SLA_attribute, a proposal must consist of a complete SLA. This provides each agent with a context for the subsequent exchange of counter-proposals.

- **(propose** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩ ⟨SLA_object_id⟩)

⟨agent_id⟩ is the unique identifier of the agent that is to receive the proposal or the agent from which the proposal was received.

⟨service_name⟩ is the name of the service that is referred to in the proposal.

⟨conversation_id⟩ is a unique identifier generated by the proposing agent that distinguishes the conversation initiated by this proposal from any other negotiation strands being pursued.

⟨message_id⟩ is the identifier of the particular message within this conversation. Using this and the conversation identifier, an agent can keep tract of the progress being made during a particular negotiation.

⟨SLA_object_id⟩ is the INSTANCE NAME of an instance of the object class Adept_Sla.

counter-propose: The counter-propose interface enables agents to exchange modifications to the initial proposal. The modifications suggested by an agent will be at least one SLA_attribute. For example, an agent may counter-propose a lower price, or a higher price as well as a higher volume. Note that in terms of types the final field of a counter-propose message is identical to that of a propose message. However, a proposal will necessarily contain a whole SLA, and a counter-proposal may contain any non-empty part of an SLA.

⟨SLA_attribute⟩ is the slot name of an attribute of an SLA. This is used to inform the CM of the slots that have been changed, and hence are being counter-proposed.

- **(counter-propose** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩ ⟨SLA_object_id⟩ ⟨SLA_attribute⟩+)

accept: The accept interface enables agents to send and receive messages accepting an SLA, and hence completing a negotiation with an agreement.

- **(accept** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩)

reject: The reject interface enables agents to send and receive messages rejecting an SLA, and hence completing a negotiation without agreement.

- **(reject** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩)

refuse: The refuse interface.

- **(refuse** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩)

failure The failure interface.

- **(failure** ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩)

inform The inform interface.

(inform ⟨agent_id⟩ ⟨service_name⟩ ⟨conversation_id⟩ ⟨message_id⟩)

[0058] There follows a description at the design of the agent's internal negotiation deliberation mechanisms. Each agent in the scenario is assumed to be architecturally equivalent. Sections below are the detailed description of the components of this architecture.

[0059] The reasoning model determines the agents behaviour in a given negotiation context. It is responsible for

- initiating negotiation to obtain a desired service
- responding to proposals from other agents
- determining when proposals should be accepted or rejected
- and determining when counter-offers should be made and what these counter offers should be.

[0060] Negotiation has three reasoning components (see [1] for a formal specification) which are supported by information maintained in the agent models and the agent's working memory.

- The *evaluation* reasoner takes proposals or counter-proposals coming in from other agents and determines whether they should be accepted, rejected or whether a counter-proposal should be generated. If a counter-proposal is appropriate, control is handed to the strategic and tactical reasoners to produce a response.
- The *strategic* reasoner decides, at a coarse level of granularity, how the agent should approach the particular negotiation. For example, whether it should be co-operative or competitive, whether time or resources is the primary consideration, etc.
- Finally, the *tactical* reasoner fills in the slots of the SLA in a way that enacts the chosen strategy.

[0061] The reasoning components have two main repositories for information: the working memory (WM) and the agent models. The former represents transitory information related to ongoing negotiations, while the latter represents persistent storage of more stable information.

[0062] Information stored in the working memory is structured around the notion of a *negotiation thread*. A thread is essentially a record or history of utterances related to a particular negotiation need (i.e. finding a server for a particular service). It includes all the messages the agent has sent, all the messages the other agents have sent, which strategies and tactics the agent has deployed, the current status of all negotiation threads (in cases where the agent is managing multiple threads of negotiation for the same service), and the services earliest start and latest end times. Management of and traceability of concurrent threads is an important factor in many party negotiation. In order to achieve this an agent maintains a unique thread identifier with other agent/s using the following information:

(WM:: agent_id: Type Symbol
conversation_id: Type Symbol
message_id: Type Symbol
SLA_object_id: Type Symbol)

where the slots represent:

- ⟨agent_id⟩ is a unique identifier of the agent engaged in negotiation
- ⟨conversation_id⟩ is the unique conversation identifier

- ⟨message_id⟩ is an identifier for marking a number of message exchange
- ⟨SLA_object_id⟩ is an INSTANCE-NAME that points to a SLA object.

[0063] In the context of supporting negotiation, the agent models represent the agent's (private) beliefs about itself and its environment. The acquaintance model (or AM) is the storage site of the information an agent knows about other agents and is represented as the structure:

```
(AM::  agents : Type Symbol
       topology : Type Symbol
       status : Type Symbol
       capacity : Type Symbol
       protocol : Type Symbol)
```

where the slots represent:

- agency agents: unique names of individual members of the agent's agency;
- agency typology: the agent's relationships (peer, subsidiary agent, etc.) with other community members;
- agency status: which agents are in the same organisation and which are external;
- agency capacity: which agents can provide services the agent needs
- the negotiation protocol; the rules of interaction

[0064] The self model (or SM) is the storage site of the information an agent knows about itself and can be represented as:

```
(SM::  service: Type Symbol
   issues: Type Symbol
       reservation-values: Type Integer
       weights: Type Integer
       preferences_orderings/utility(): Type Symbol
   resources: Type Integer

           Clock: Type Integer
           commitments: Type Symbol
           tactics: Type symbol)
```

where the slots represent:

- service descriptions for the services the agent can perform itself together with an indication of the number of concurrent invocations which are permissible;
- the set of issues in the negotiation set. Note, that for issue extension we will assume that all agents have a representation (or common ontology) for all possible set of issues as well as the associated reservation values, weights and utility functions.

- the agent's reservation values for each issue in negotiation for the services it consumes and provides;
- importance of various issues under negotiation
- the preference orderings (or utility function) for the ranges of the issue.
- a representation of time
- the commitments the agent has already made through its SLAs;
- and the set of available tactics the agent can use to compute offers

[0065] The evaluation reasoner becomes active when an agent receives a proposal or counter-proposal from another agent. Upon receipt of such a message, the agent computes the utility it attains for the proposal. It uses an additive scoring function over each slot in the SLA where each slot is assigned a weight representing the relative importance of that issue to that agent. For example, when an agent receives a SLA it goes through each slot in the proposal and computes a measure of desirability (a utility rating between 0 and 1) to the value contained therein. The raw utility values are then multiplied by a weighting factor (which indicates their relative importance) and then summed over all the slots. This process produces a single utility value for the proposed SLA. In parallel, the agent sends the offer just received to the tactical reasoner to see what offer the agent would produce next using its current strategies and tactics. Once computed, this offer is returned to the evaluation reasoner and rated using the aforementioned scoring function. If the utility of the offer the agent would have sent is less than the utility of the offer just received, the offer is accepted. Acceptance involves a conditional commitment by the server that it will execute the specified service under the SLA's terms and conditions. The commitment is conditional in that the client still has to confirm or deny the contract. Assuming the client confirms the contract, it then terminates all other negotiation threads for the same service instance. The second outcome of the SLAs evaluation is that the proposal is rejected. This occurs when: (i) the deadline for reaching an agreement has been reached; or (ii) another agent has been selected to perform the service. The final evaluation outcome is that the offer is neither accepted nor rejected. In this case, the agent generates a counter offer.

[0066] If a counter offer is to be made, the evaluation reasoner also makes an assessment of the opponent's negotiation behaviour in the current thread with respect to time. Thus evaluation is not only confined to the current offer instance, it also incorporates the relationship of that offer to previous ones in the thread. In particular, the agent classifies the behaviour of its opponent into one of three mutually exclusive states: i) CONCEDING: the utility to the recipient of the last offer is greater than the previous offer received from that agent; ii) EXPLOITING: the utility to the recipient of the last offer is less than the previous offer received from that agent; or iii) STALEMATE: the utility to the recipient of the last offer is the same as the previous offer received from that agent. As well as the direction of change, the agent uses the negotiation thread history to determine the rate of change of that state. Thus the agent calculates whether this conceding/exploiting is INTENSIFYING, LESSENING, or CONSTANT. These two pieces of information are then passed onto the strategic reasoner which uses them to determine whether its present strategy is being successful or whether a change is needed. The pseudo-code for the evaluation reasoner is given below.

```
evaluate(Propose OR Counter-propose, SLA_in){

   issues(Weights),

   /* Retrieve preferences over the issues */

   thread(Thread),      /* Retrieve the Thread */
```

```
utility(SLA_in,Weights,Value_in),

/* compute the utility of the offered SLA */

compute_new_offer(Strategy,Tactic,SLA_out),

/* compute what offer would have been generated */

utility(SLA_out,Weights,Value_out),

/* compute the utility of that offer*/

if Value_out < Value_in then accept,

else time_left=0, then reject,

else have_SLA then reject,

else counter_propose(SLA_out),

        negotiation_state(Thread).

        /* compute the state of negotiation */ }
```

[0067] The strategic reasoner is invoked by the evaluation reasoner in the case of an ongoing negotiation or by request for new negotiations. In either case, the purpose of the reasoning at this level is to set broad guidelines about how the agent should behave in a particular negotiation context. These guidelines relate to determining the relative importance of the three classes of behaviour which take time, resources, and an opponent's behaviour as the primary basis for computing an offer. Time is important when the negotiation has a deadline. Resources need to be considered so that the agent expends an amount appropriate to the value of the contract. The opponent's behaviour is considered to ensure the agent is not exploited during the negotiation. The relative importance of the three classes is expressed by assigning a series of weights to the alternatives.

[0068] For new negotiations, the agent receives information about when the service is required (HAVE-TIME, NOW), uses AM information about the number of known suppliers of the service (ONE, MANY), and uses AM information about the agent's relationship with the potential service provider (SAME-ORGANISATION, EXTERNAL-ORGANISATION) to set the strategy.

[0069] The first strategic decision relates to the logistics of the negotiation: who to negotiate with, whether to negotiate with more than one agent, and if more than one agent is to be negotiated with then should the negotiation proceed sequentially or in parallel. If there is only one service provider then the agent has no real choice to make at this level.

[0070] Having decided upon the logistics, the agent must determine how it is to behave. In addition to setting the strategy, the agent records its expectation of how the negotiation will develop in terms of the speed at which it will converge and the likely response of the opponent. This information is then used to monitor the progress of the ongoing negotiation.

[0071] For ongoing negotiations, the role of the strategic reasoner is to determine whether the current strategy is being successful (in terms of the agent's predictions about its development and in terms of the utility the agent is obtaining from the deal) in fulfilling the agent's negotiation objectives. Such monitoring is needed because the world in which the agent is operating is subject to change (e.g. the agent may require the service sooner/later than it estimated or a new provider for the service may be discovered) and also because operating a fixed, unchanging strategy means the agent is more open to exploitation by its opponents (since its behaviour is easier to predict). Strategy modification is triggered by two types of event (i) whether there is a change in the agent's internal state (e.g. whether the time by which

an agreement should be in place is becoming critical); and (ii) how the opponent is behaving (e.g. CONCEDING, EXPLOITING, STALEMATE, INTENSIFYING, LESSENING, CONSTANT). The pseudo-code for the strategic reasoner is given below.

```
begin(strategy){
   if(Negotiation=new) then{
         compute(behaviour_importance),
         /* assign weights to different tactic classes
*/
         if(agents > 1) then


            compute(logistics),
            /* compute who to negotiate with and when*/
            compute(expectations)
            /* compute likely response of strategy */}
      else assess(goals)
            /*assess distance to objective */ .
   end(strategy)}.
```

[0072] The role of the tactical reasoner is to enact the high-level behaviour set by the strategic reasoner. The output of this level is a SLA which has values in each of its slots. Thus a tactic is a function which acts in line with the set strategy, to set a value for each SLA slot. For quantitative slot parameters, tactics have to select a value in between the allowable minimum and maximum value for that issue. For qualitative values, the tactics have to choose from a discrete range of alternatives; a process achieved by mapping the qualitative values onto the quantitative scoring function.

[0073] The way in which tactics differ is in how they go about computing a slot value. Them are three main ways of coming to a value.

Time-dependent tactics: This family of tactics base their behaviour on the time remaining until an agreement must be in place. At their negotiation deadline all these tactics put forward their reservation values. However the way in which they concede to reach these values differs. There are two broad patterns of concession: (I) *boulware*: maintain the offer until the time is almost exhausted and then begin to concede up to the reservation value; and (ii) *conceder*: move rapidly to the reservation value.

Resource-dependent tactics: This family of tactics base their behaviour on the amount of a given resource remaining. The property of these tactics is that they model the urgency of the deal as: i) the resources become scarcer, ii) the willingness of other parties in negotiation decreases (measured as an increase in the length of the negotiation thread) and iii) the computational load on the agent increases. The actual relationship is that the quantity of time left in negotiation is directly proportional to the number of agents in the negotiation and inversely proportional to the length of the negotiation thread. Thus, the more agents who are potentially available to perform the service, the

longer the agent can afford to negotiate. But the longer the duration of the negotiation, the more urgent the need for an agreement becomes.

Behaviour-dependent tactics: This family of tactics base their behaviour on how their opponent behaves during the ongoing negotiation thread. The tactics within this family differ in which aspect of their opponent's behaviour they imitate, and to what degree. There are three ways in which behaviour can be imitated: i) Relative Tit-For-Tat; ii) Absolute Tit-for-Tat; and iii) Averaged Tit-For-Tat, where others behaviour is, respectively, imitated proportionally, absolutely and in an averaged fashion.

[0074] Each of the families computes a value for each of the negotiation issues based upon their particular perspective. The three values for each issue are then combined, according to the relative weightings set by the strategic reasoner, to provide a single value which is the one put forward for that issue.

```
begin(tactic){
   foreach(issue){
         compute_offer(time-dependent,Of1),
         compute_offer(resource-dependent,f2),
         compute_offer(behaviour-dependent,Of3),
         /* compute offer for each issue using all
tactics */
         +((Of1*time-dependent_weight),
              (Of2*resource-dependent_weight),
                   (Of3*behaviour-dependent,Weight),
         /*combine the offers according to their weights
*/
         instantiate(SLA),}

                        end(tactic)}.
```

[0075] The developed model described above does not support negotiation over qualitative issues since it is based on quantitative computation. Tactics functions have a range of values (the reservations) and a single environmental variable as their domain and a discrete value (an offer for an issue) as their range. Moreover, the rate of change of these offers are continuous.

[0076] The proposed solution is to represent reservation values differently by mapping each attribute of a qualitative issue into a value system and then computing offers not using the reservation values of the issue (i.e. the actual ranges of the min_j, and max_j) for qualitative issue j but with the *values* of the issues. For example, consider the case where an issue has three attributes: blue, red and green. The first step in negotiation over qualitative issues is to map these attributes into values by assigning to each attribute a value. Note, that we must have ordered preferences.

[0077] More formally, let $D_j = (q_1, q_2, \ldots q_n\}$ be the set of qualitative attributes, where q_x is attribute x of the quali-

tative issue (e.g. blue). Then, $min_j = min(V_j(D_j))$, and $max_j = max(V_j(D_j))$. The next step is to redefine elements of tactics. Assume that it is the turn of agent *a* to utter a new offer. Given this new representation of reservation values the time-dependent and resource-dependent tactics become:

$$X_{a \to b}[j] = \begin{cases} inverse(min_j + \alpha(t)\,(max_j - min_j) & \text{if } V_j \text{ is decreasing} \\ inverse(min_j + (1-\alpha(t))\,(max_j - min_j) & \text{if } V_j \text{ is increasing} \end{cases}$$

where $\alpha(t)$ is computed as before (for time-dependents based on the difference between t and t_{max} and for resource-dependents as the amount of resources) and inverse() is the function that remaps the value (say 0.6) to a discrete qualitative attribute (say red). The form of inverse() function is given below.

[0078] Likewise the imitative tactics become:

$$X_a \to b[j] = \begin{cases} min_j & \text{if } P \le min \\ max_j & \text{if } P \ge max_j \\ Inverse\,(P) & \text{otherwise} \end{cases}$$

where

$$P = v_j\,(x_{b \to a} \wedge t_{n-2\delta}[j]) \,/\, v_j\,(x_{b \to a} \wedge t_{n-2\delta+2}[j]) * v_j\,(x_{a \to b} \wedge t_{n-1}[j])$$

[0079] The inverse() function is the nearest neighbour algorithm which remaps quantitative values of utility / value of a deal back into the nearest attribute of the qualitative issue. The pseudo-code for the nearest neighbour is:

```
0 compute_distance(V(x_j),V(D_j)),
                                         /*compute distances to values of
                                         attributes in the domain of attributes
                                         given the value of the offer */
1 if(equidistance) then randomly select
2 else select(closest)          /*select the closest v(d_j) to
v(x_j) /*
3 remap(V(x_j),attribute)       /*map attribute value back to
attribute label */
```

[0080] Because the final offer is the product of what the tactics suggest for the issue and what the weights associated with that issue (i.e multiplication of a qualitative and quantitative values respectively) we will model the inverse() function at the strategic level. Therefore, the tactics suggest a numeric output for the value of an issue and only when this value is multiplied by the appropriate weight will the inverse() function produce the remapping into attribute label.

[0081] The introduction of new issues has been identified as an advanced ANSAP scenario, where agents can dynamically introduce new issues into the negotiation set through the negotiation thread. One reason for this introduction is that agents may decide to increase the likelihood of convergence of a negotiation thread. Agents are therefore required to:

- recognise negotiation states which can lead to possible convergence problems
- and reason about possible solutions

[0082] The proposed solution for the recognition problem is to dynamically track the rate of change of the utility of the offered SLAs. The rate of change of utility can be determined by determining two parameters (which is given by the user) : i) the length of the thread which will be used for determining the state of the negotiation thread (call this *L*) and ii) a threshold which can be used to classify the thread states (call this (Θ). The agent in effect compares the utility of the last offered contract with that of the contract L steps back in the history of the negotiation (or thread) and ascertains whether the differences between the overall utility of these two contracts are within a given threshold. If they are then a divergence from the required dynamics is detected and a new issue is added to the current negotiation set. Otherwise negotiation continues as usual.

[0083] The combined deliberation mechanism is given by the pseudo-code:

```
Evaluate_state(Thread,L,θ,State){ /*evaluate the state of the thread given L and θ */
Utility(last_offer,X), /* compute utility() of last offer */
utility(Thread,L,Y)},/*compute utility of previous offer L steps back */
If(abs(X-Y) > θ) then Introduce(Issue)/* introduce issue of choice if θ is reached */
  Else continue
```

[0084] The choice of which issue to Introduce is domain dependent and agents are assumed to know which issues can be included. For simplicity the current implementation will select the first issue that is on top of the stack, although future implementations can be more intelligent by selecting the issue which has the lowest weight first.

[0085] Introduction of a new issue has direct influence on the management of threads and the agent's knowledge-bases. In particular, behaviour-dependent tactics which compute an offer for an issue based on history of offers have to compute the values for a new issue which does not exists in the thread history. Therefore, we will assume that these tactics continue to concede until there exists enough utterances over the new issue (in the window given by δ) to compute their values.

[0086] Introduction of a new issue also influences the beliefs of not only the agent that introduces an issue but also the agent who is offered a new issue. Both agents must update their self models over: i) the set if issues, ii) data relevant to the new issue (such as the reservation values, κ and utility function information), iii) the new set of weights given the new issue (given the constraint that all weights must normalise to 1) and iv) the values of τ over the new issue (or the strategy the agent adopts over the new issue).

[0087] Furthermore, we will assume that all agents have a representation of all the possible set of negotiation issues involved in the domain of discourse (a common ontology). However, in cases where this is not the case (i.e. the new offered issue is not a member of the representational set of issues in the receiver's set) then the receiver agent must have the capability of: i) delete the last utterances from the thread and ii) responding with a message that informs the sender agent of the ontological problems. The proposing agent can then select the next issue from the new issues which is on top of the stack. If no other new issues are available then the agent continues negotiation with the existing set of issues.

[0088] Finally, since the rationale for introduction of a new issue is to "kick start" / escape local minima in negotiation, then the agents must have conciliatory attitudes towards the newly added issues. Therefore, κ must be high, τ set to high values over conciliatory tactics and low weights over the new issue (to reflect their lower significance than "core" negotiation issues).

[0089] Any range or device value given herein may be extended or altered without losing the effect sought, as will be apparent to the skilled person for an understanding of the teachings herein.

Claims

1. A method of negotiation, over a set of issues, between automated entities comprising the step of:

   modifying the set of issues during the negotiation.

2. A method according to claim 1 wherein the step of modifying the set of issues comprises the step of:

   adding an issue to said set of issues.

3. A method according to claim 1 wherein the step of modifying the set of issues comprises the step of:

removing an issue from said set of issues.

4. A method of negotiation over a set of issues, between automated entities comprising the step of:

at least one of the entities making a counter-offer wherein its score for at least one of said issues is lowered and its score for at least one of said issues is raised.

5. A method of negotiation over a set of issues comprising at lest one qualitative issue, and comprising the steps of:

mapping the values of said qualitative issues to qualitative values;

negotiating using said qualitative values.

6. A method according to claim 5 comprising the steps of:

mapping from said qualitative values to said qualitative on completion of negotiation.

7. Apparatus arranged to perform a method according to claim 1.

8. Software on a machine-readable medium embodying the method of claims 1.

9. Software on a machine readable medium arranged to perform the method of method of negotiation over a set of issues, between automated entities, the method comprising the step of at least one of the entities making a counter-offer wherein its score for at least one of said issues is lowered and its score for at least one of said issues is raised.

10. Software on a machine readable medium arranged to perform the method of method of negotiation over a set of issues comprising at lest one qualitative issue, and comprising the steps of mapping the values of said qualitative issues to qualitative values and performing negotiation using said qualitative values

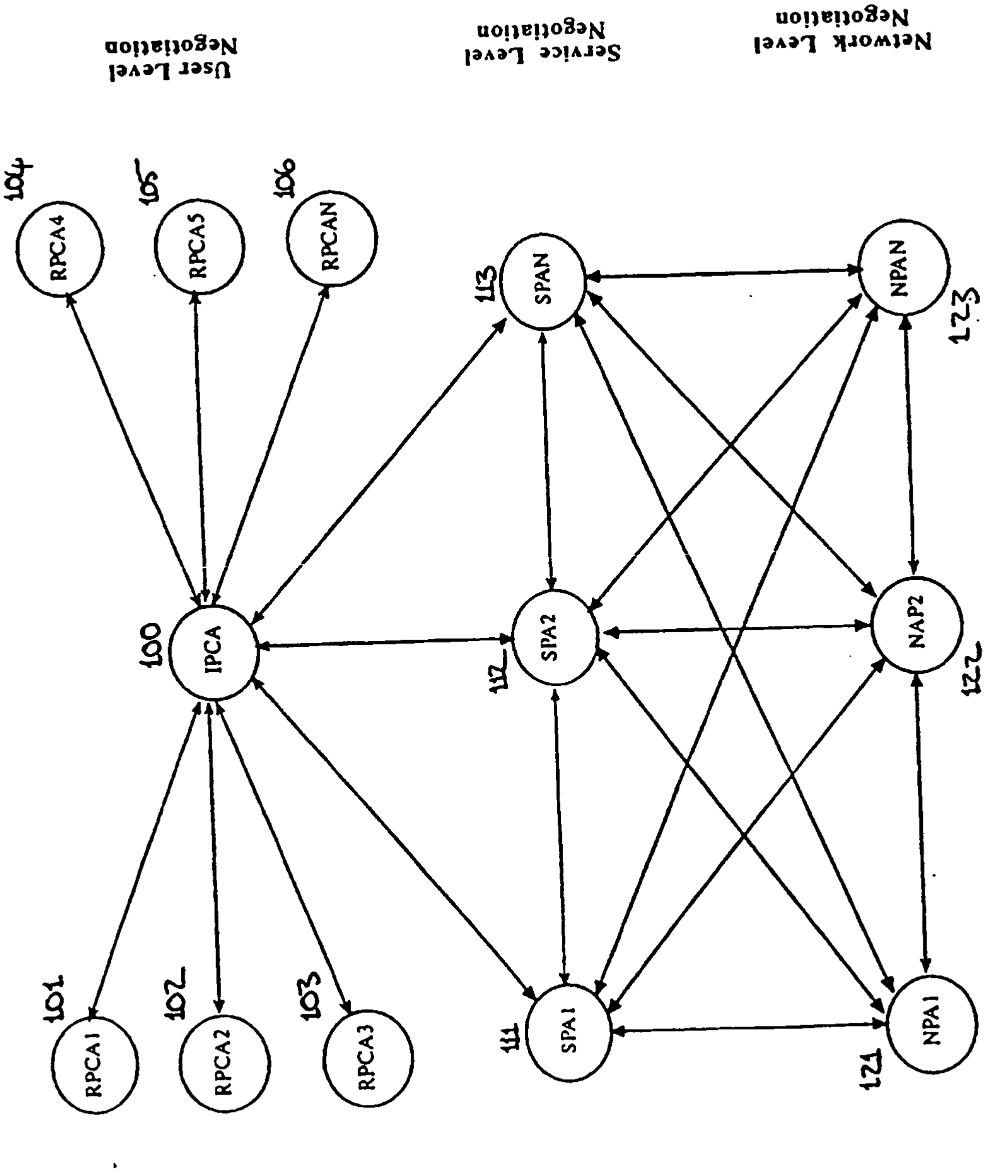

User Level Negotiation
Service Level Negotiation
Network Level Negotiation
RPCA1
RPCA2
RPCA3
RPCA4
RPCA5
RPCAN
IPCA
SPA1
SPA2
SPAN
NPA1
NAP2
NPAN
100
101
102
103
104
105
106
111
112
113
121
122
123

Fig. 1

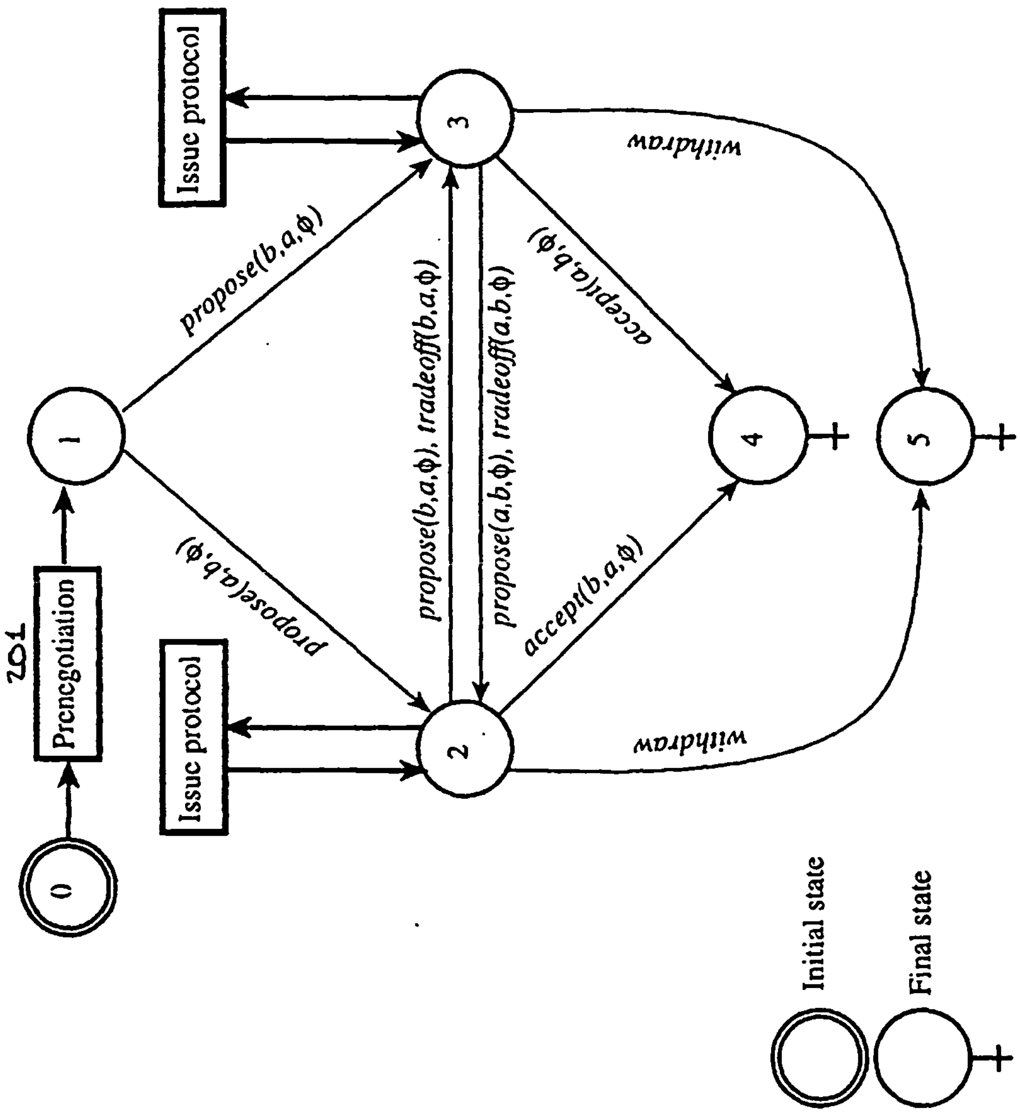
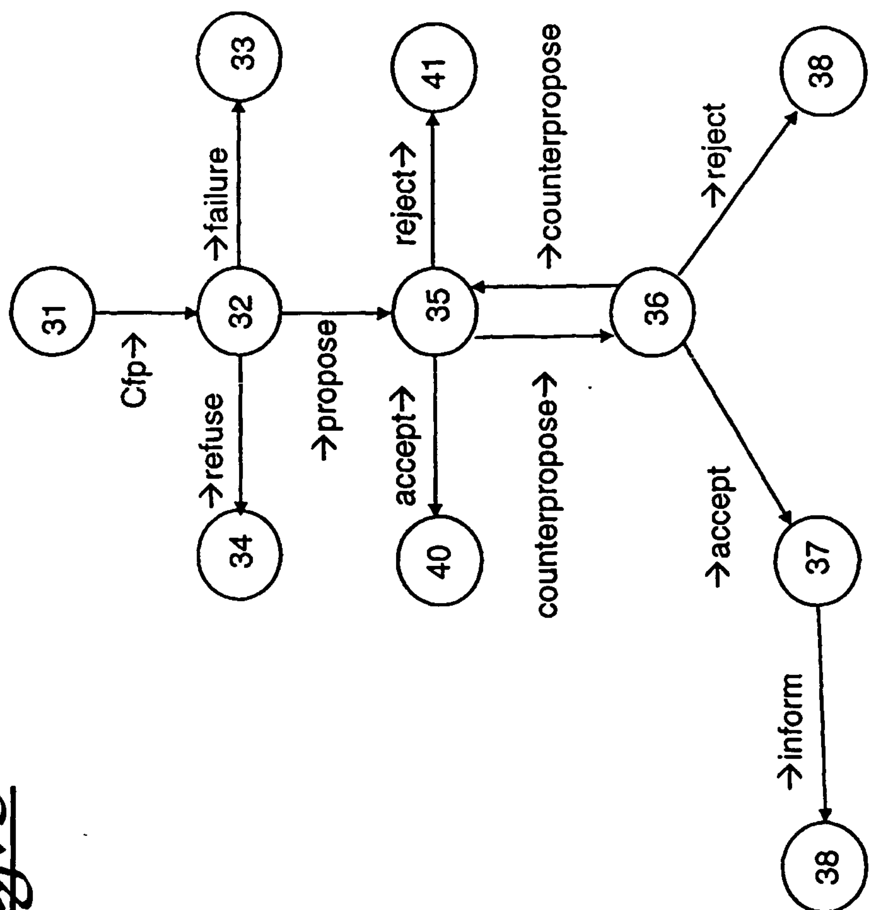

0
Prenegotiation
201
1
propose(b,a,φ)
propose(a,b,φ)
Issue protocol
3
Issue protocol
2
propose(b,a,φ), tradeoff(b,a,φ)
propose(a,b,φ), tradeoff(a,b,φ)
accept(a,b,φ)
accept(b,a,φ)
withdraw
withdraw
4
5
Initial state
Final state

Fig. 2

Fig. 3


31
Cfp→
32
→failure
33
→refuse
34
→propose
35
reject→
41
accept→
40
→counterpropose
counterpropose→
36
→reject
38
→accept
37
→inform
38

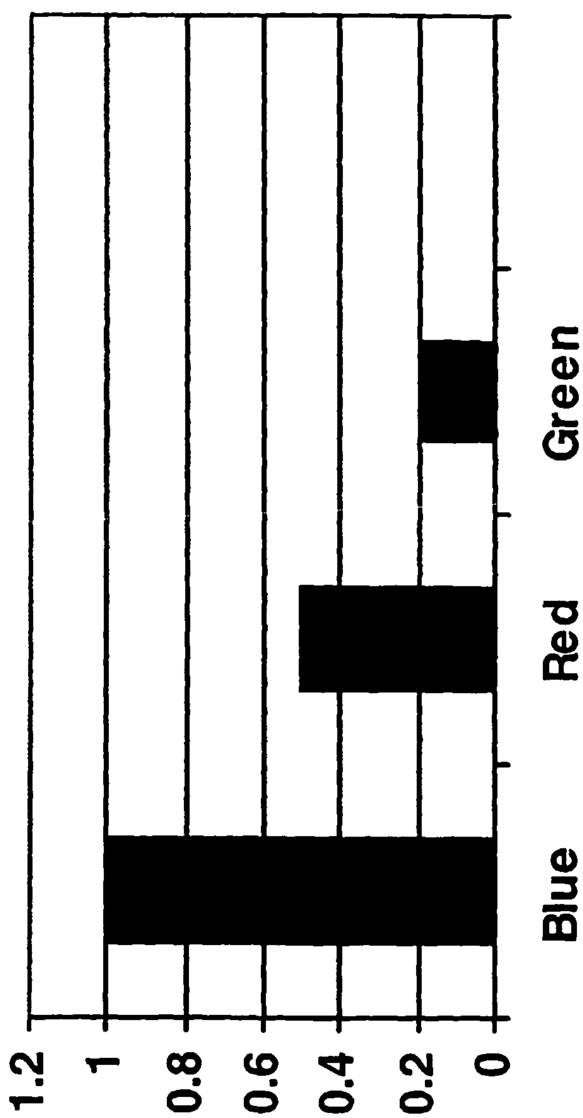


Fig. 4